APPLICATION

FOR

UNITED STATES LETTERS PATENT

APPLICANT NAME R. G. Hartman, et al.

TITLE System and Method for Enabling Graphic Applications In an Interactive Programming Model

DOCKET NO. END9 2000 0050 US2

INTERNATIONAL BUSINESS MACHINES CORPORATION

CERTIFICATE OF MAILING UNDER 37 CFR 1.10

I hereby certify that, on the date shown below, this correspondence is being deposited with the United States Postal Service in an envelope addressed to the Assistant Commissioner for Patents, Washington, D.C., 20231 as "Express Mail Post Office to Addressee" on 12/21/00

Mailing Label No. EL598673183US

Name of person mailing paper: Denise M. Jurik

Denise M Jurik 12/21/00

Signature Date

a:formal2.lwp

SYSTEM AND METHOD FOR ENABLING GRAPHIC APPLICATIONS IN AN INTERACTIVE PROGRAMMING MODEL

Related Applications

This application is a continuation-in-part (CIP) of U. S. Patent application S/N 09/574,138 filed 18 May 2000 for "System and Method for Enabling Graphic Applications in an Interactive Programming Model".

Background of the Invention

Technical Field of the Invention

This invention pertains to client/server systems. More particularly, it relates to enabling graphical applications to run on a traditionally text based host.

Background Art

Many new software applications developed for the Internet are Java based, because Java is a prime operating system (OS) portability facilitator. Theoretically, a

Java-based application can run on any Operating System unchanged, be it Windows95, Linux, OS/2, VM, etc. It also has the advantage of being able to tie into the Graphical User Interface (GUI) support of a host through an Abstract Window Toolkit (AWT), which is a standardized OS interface for application graphics support. (AWT is a product of Sun Microsystems, and is further described at http://java.sun.com/products/jdk/awt/.)

Having GUI support enables the blending together of graphics and text from a variety of sources into one seamless screen or "panel", a highly desired attribute for any OS or application.

Unfortunately, many legacy application environments don't conveniently lend themselves to a GUI interface. Most of these legacy machines have added additional support consistent with the connectionless, stateless, http class of web servers. These Web based applications have to incorporate some method of state maintenance in order to mimmick a traditional connection and state oriented legacy programming model. These models are typically mainframe interactive. Many legacy applications can't really blend graphics and text very well -- that is, until recent technology innovations have afforded some relief. For

example, through use of Java applications great strides have been made in blending graphics and text.

IBM's AS/400 system legacy programming model has a text-only "green screen" interface (so named after the default color of text on the screen). This presents a problem porting popular Java applications to the AS/400, since many such applications require GUI support. Since the AS/400 system is a business computer, this means many businesses cannot easily migrate their legacy applications to their customers with a GUI interface. For example, suppose company ABC, Inc. offers customers terminal access to its AS/400 system and all the business applications installed on it, charging an hourly rate for connect time to use these applications. Customers connect to the AS/400 using a Telnet Client and get a text-only terminal emulator, at best. They cannot take advantage of GUI enabled Java applications that may be installed or ported over from other platforms.

IBM solved this problem on the AS/400 by adding Remote Abstract Window Toolkit support (RAWT). With Remote AWT, Java AWT graphical programs can be executed on a AS/400 system while remotely displaying the graphics.

The use of Remote AWT requires that Transmission Control Protocol/Internet Protocol (TCP/IP) be set up, and Sun Microsystems, Inc., Java Developer's Kit (JDK) 1.1.x installed on both the server AS/400 and the remote display. Any graphics-capable hardware, including IBM Network Station, can be used as a remote display for Remote AWT as long as it includes:

1) Graphics-capable hardware that runs Windows 95, Windows NT 4.0, Windows ME, Windows 2000, IBM Operating System/2 (OS/2), Sun Solar, Sun Solaris or AIX.

2) Configured hardware to access AS/400 with TCP/IP.

3) Java Developer's Kit 1.1.x (JDK 1.1.6 or later is recommended.)

AWT refers to Abstract Window Toolkit. The AWT is part of the Java Foundation Classes (JFC)-- the standard API for providing graphical user interfaces (GUIs) for Java programs. This is a platform-independent windowing, graphics and user interface toolkit. With the Remote Abstract Window Toolkit (RAWT), a Java AWT graphical program can run on the AS/400 (a text-only platform) and display the

graphics remotely. To use Remote AWT, the Transmission Control Protocol/Internet Protocol (TCP/IP) is set up, and Sun Microsystems, Inc., Java Developer's Kit (JDK) 1.1.x installed on the AS/400 and remote display.

As an example, this support is used to supply the interface for Operations Navigator, a remote configuration tool that ships free with every AS/400 as part of the 5769-XE1 Licensed Program Product, and allows AS/400 System Administrators to configure an AS/400 using a remote Windows 95/NT/2000 platform.

There are also Java-based terminal emulator clients, such as IBM's Host-On-Demand product. These are truly GUI clients, but they connect to the traditional TCP/IP Telnet Server, which primarily supports "green screen" applications. IBM's Network Station is similar, in that it is also a GUI capable client and can exploit tunneling of graphics from the integrated file system (IFS) on the AS/400, but this is not using Java virtual machine (JVM) capabilities on the AS/400. Further, Host-On-Demand must do many kinds of datastream conversions to work with a variety of Telnet Servers. For example, to communicate with OS/400 (AS/400), the Java client must convert 5250 datastreams into

something the host GUI understands, in order for it to display on the client Operating System. Likewise, for VM systems (S/390) 3270 datastreams must be converted to host GUI. Replicating this across a few more platforms results in considerable code expansion in a Java-based client, if it is expected to support more than one platform.

Host-On-Demand supports "servlets", which are supposed to be Java applets that run on a server machine. But, by requiring a Host-On-Demand client, servlets cannot be exploited on Thin Clients, such as Network Stations.

The problem with using Operations Navigator to run Java applications is that it is a custom application, and requires a custom server on the AS/400 system to run the Remote AWT. There is no Internet standard or protocol by which graphical clients can connect to the AS/400 system and run the Remote AWT (and by extension, a Java application).

It is, therefore, an object of the invention to provide an improved system and method for enabling graphics enabled applications to run on a text based host.

It is a further object of the invention to provide a

system and method whereby an AS/400 host can enable standard Telnet Clients to connect and be able to receive output from Java graphics applications.

It is a further object of the invention to provide a system and method for graphics enabled application platform independence for workstations by supporting a variety of clients and hardware.

It is a further object of the invention to provide an improved system and method for exploiting a Java virtual machine on a text based host system to run both text based and remote graphics applications, such as Java and X Windows applications.

It is a further object of the invention to provide a system and method for enhancing the ability of a text based host system to perform work management, including authentication, NLS, and job routing, for many clients at once.

It is a further object of the invention to provide an improved system and method for centralizing applications and support for applications.

It is a further object of the invention to provide an improved system and method for centralizing of backup/recovery processes.

It is a further object of the invention to provide an improved system and method for centralizing upgrades/fixes, such that such upgrades and fixes need be done only one time, not once for each workstation.

It is a further object of the invention to provide a system and method for comprising a single source for consulting, leasing, and marketing text based and graphical applications.

It is a further object of the invention to provide an system and method for supporting thin clients, such as network stations, by offloading CPU cycles workstations to a central mainframe.

It is a further object of the invention to provide an improved system and method for using existing terminal emulators which requires no new development and exploits existing standards, including Internet RFC'S.

Summary of the Invention

In accordance with the system and method of the invention, a multimedia enabled application runs on a text based host. A client negotiates a connection with a server on a first port, and informs the server that the client is multimedia enabled and is listening on one or more additional ports for multimedia application data. Responsive thereto, the host establishes a multimedia connection from a virtual machine executing a selected application to the second port on the client for presentation of a multimedia application interface at the client.

Other features and advantages of this invention will become apparent from the following detailed description of t he presently preferred embodiment of the invention, taken in conjunction with the accompanying drawings.

Brief Description of the Drawings

Figure 1 is a system diagram illustrating the system of

the invention for enabling graphics applications to run on a text-based host.

Figure 2 is a flow diagram illustrating the steps executed at a client workstation to initiate a graphics session with a text-based host server.

Figure 3 is a flow diagram illustrating the steps executed at a text-based server responsive to a request from a graphics capable client for a graphics session.

Figure 4 is a system diagram illustrating a more detailed, exemplary embodiment of the system of the invention.

Figure 5 is a flow diagram illustrating in greater detail the method steps executed by a text-based host server according to the exemplary embodiment of Figure 4.

Figures 6A-6B are a flow diagram illustrating the session negotiation protocol of the exemplary embodiment of the invention.

Best Mode for Carrying Out the Invention

In accordance with the invention, graphics enabled applications run on a text-based host server by allowing a client application running at, for example, a workstation to inform the server of this session (1) that it, the client application, is graphics capable and (2) the IP address and port(s) it is waiting on; and then by having the server set capability indicia, such as RAWT attributes, in the operating system for this session to indicate the (1) the client is graphics enabled, (2) the IP address and port(s) it is waiting on, (3) optionally, the path to an application to be automatically launched.

Depending upon the context, the terms "graphics" and "multimedia" are intended to encompass all forms of non-text data.

Referring to Figure 1, a preferred embodiment of the system of the invention is shown. Host server system 100, such as an IBM AS/400 (TM) or System/390 (TM) system, includes a graphics/multimedia capable application library 104 and a text based application library 109, one or more

virtual machines 106, one or more devices, such as graphics enabled devices 108 and text enabled devices 110, interactive subsystems 114, 116, workstation server 118 including sockets 102 interface to network 130, and windowing toolkit 112. Workstation 120 includes a display device capable of presenting GUI window 122, a Program Information File (PIF) 124, a graphics client application 126, a telnet-based program launching tool 128, ports 134 and 136, sockets 132 interface to network 130.

A launching tool, as used herein, refers to an a selectable element, such as an icon or entry in a pull-down menu, or the like, with associated information. The selectable element represents the program to be launched remotely on the server system. When a user at the client selects, such as by clicking on, the element, the launching tool obtains from, for example, a file the program name, directory, RAWT port -- all information that is required at the server side to start the program. That information is passed to the server over the telnet connection. At the server, the program is loaded and executed.

Referring to Figure 2, workstation 120 executes the

following. In step 400 telnet-based program launching tool 128 negotiates from a given IP address and port 136 a connection to server system 100, negotiates for graphics support, and negotiates for a number of ports. In step 404, telnet-based program launching tool 128 informs workstation server 118 of the port 134 on which graphics client 126 is listening. In step 406, client 126 listens on GUI capable port 134. In step 408, client 126 receives a GUI connection from server 118. In step 410, the GUI is processed by graphics client 126 and displayed at GUI window 122. In accordance with this embodiment of the invention, a program that initiates remote execution from the client, rather than a full telnet client emulator, is all that is required.

Referring to Figure 3, server system 100 executes the following. In step 420, server 118 negotiates a connection with telnet-based program launching tool 128 at a given IP address and port 136. In step 428, responsive to receiving in step 424 from telnet-based program launching tool 128 indicia specifying that workstation 120 is graphics capable and the port 134 on which graphics client 126 is listening, server 118 sets capability indicia in device space 108. In step 428, system 100 determines whether it will proceed with graphics processing and, if so, in step 432 initiates a virtual machine 106, and if not, the connection is closed.

In step 430, virtual machine 106 connects through windowing toolkit 112 to port 134 of graphics client 126 and graphics window 122 on behalf of a selected application from library 104. If a virtual machine 106 is initiated, graphics capable applications from library 104, are processed by graphics enabled device 108, with communication to workstation 120 port 134 via windowing toolkit 112. If a virtual machine is not initiated, the connection is closed.

Referring to Figure 4, in accordance with a preferred, more specific embodiment of the invention, by using the existing TN5250E support, Telnet Clients configured with Remote AWT 326 (the requirements for which are discussed above) are enabled to negotiate a graphical Java run-time session over port 334 from any graphical workstation 320.

Referring to Figure 5 in connection with Figure 4, in operation, workstation server, such as an IBM AS/400 or System/390 Telnet Server 318, receives from a text enabled client, such as a telnet-based program launching tool 328, a request to operate in a graphics enabled terminal-type mode. This is done by subnegotiation of user variables (USERVAR) in step 440. In this example, the USERVAR "IBMRAWTADDR", "IBMRAWTPORT", and, optionally, "IBMRAWTAPPL"

are those applicable to Remote AWT. Other graphics, or multimedia, capable applications, such as X Windows, may require negotiation of other user variables.

Referring to Figure 6, an example of step 440 negotiation is set forth. In this example a traditional Telnet handshake is done in steps 200, 202, 204, 206, 208, 216, 218, 220, 222, 224, 226, 228, 230 with Telnet server 318 to negotiate terminal-types (for a session at IP address 362 1.2.3.4 and port 336 2501) with the AS/400 Telnet Server 318 in accordance with the TN5250E Internet-draft, along with standard Internet RFC 1205 (5250 Telnet Interface), RFC 854 (Telnet Protocol Specification), RFC 855 (Telnet Option Specifications), RFC 856 (DO BINARY), RFC1091 (Terminal Type), RFC 885 (End of Record) and RFC 1572 (New Environment Option) being used. In Figure 6, IAC means "interpret as command", SB means "subnegotiation", and SE means "subnegotiation end".

In accordance with this exemplary embodiment of the invention, when Telnet server 318 receives in step 210 (Figure 6A) the three RAWT user VARS (port, address, and application), in step 444 (Figure 5) the Telnet server 318 selects a virtual terminal device 308 (herein named

QUICKnnnn 340, for example) for that session, and marks the device 308 as GUI capable. It does so by storing the values received in step 210 for the IBMRAWTADDR (IP address 360 value is '1.2.3.4') and IBMRAWTPORT (port value 334 is '2000') USERVARs into Logical Unit Device Associated Space (LUD ASP) 308 as IP address 344 and port 348, along with indicia 352, such as a bit flag, to indicate this session is operating from a GUI capable workstation 320. This GUI flag bit 352 can be used later by other processes on the host system 300, such as Work Management. In this example, the client 328 optionally requested in step 210 the application 304 QUICKEN, by sending the IBMRAWTAPPL USERVAR. In the absence of any application request, a default application such as a Java menu may be selected, which may optionally list all or some subset of available applications.

In this example, in step 448, Telnet User Exit Programs (not shown) can be configured to read the VAR and USERVAR values being sent in step 210, and select appropriate device name 340, sign-on user profile and program-to-launch values to kick off the Java application 304 requested. For example, since the application being requested in step 210 is QUICKEN, the User Exit Programs can be set to assign a free device name 340 such as QUICK0001, which will

automatically route the associated job to a dedicated sub-system QUICKEN 314. Work Management 338 manages this routing, selecting the appropriate subsystem 314, 316 when a client session is started. The User Exit Programs can further set the user profile to be QUICKUSR and the Java program-to-launch to be QUICKPGM. In this way, the User Exit Programs fully control the authority granted telnet-based program launching tool 328 to any libraries, including Java libraries 304 and text libraries 309, and programs on the AS/400 server system 300.

In step 452, responsive to a Telnet server 318 session initiation, a Virtual Terminal Manager (VTM) 306 initiates an interactive client job. Telnet server submits QUICK0001 device name 340, QUICKUSR profile and QUICKPGM program in the request sent to Work Management 338. At session initialization time, Work Management 338 initiates an interactive job using these values, bypassing the traditional, such as 5250, Sign-On panel, and launching the QUICKPGM application 304 in the Java Virtual Machine (JVM) 306. The user profile and program-to-launch can also be obtained from TN5250E negotiations rather than User Exit Programs, if host system 300 is configured that way.

Alternately, Telnet Server 318 lets Work Management 338 read the GUI bit 352 out of the QUICK0001 LUD ASP 308 and decide whether to launch a Java environment or do the default action of sending a text based, such as 5250, sign-on panel to window 324.

If a Java Virtual Machine (JVM) 306 environment is launched for the interactive session, in step 456 it will connect to the IP address 360 and port 334 extracted from the LUD ASP fields 344, 348, respectively, and establish a connection with the Remote AWT workstation client 326. Once connected, in step 460 the graphical output of the JVM environment and QUICKEN application is seen on the remote workstation in the AWT window 322. This will implement what is, in effect, a Java GUI terminal on the client workstation 320. If a plurality of ports 334 are authorized, a plurality of interactive subsystems 314 and applications 308 may be associated with those ports.

Referring again to Figure 4, Remote AWT 326 is a Remote Abstract Windowing Toolkit (RAWT) client displaying graphical and text output from Java virtual machine 306 via RAWT 312. During display via RAWT 312, the session with the telnet-based program launching tool 328 may be suspended or

hidden, and restarted upon termination application 308. Server 318 receives from workstation 320 the `RAWT VAR's and USERVAR's under RFC 1572, and selects QUICKnnnn device 308, storing GUI bit 352, IP address 344, and port 348 in LUD associated space. Subsystem 314 receives from work management the device name 340 for virtual device 308. Work management 338 sees the RAWT attribute 352 in LUD 308, and optionally initiates the Java virtual machine 306 environment. JVM 306 connects to IP address 360 and port 334. Output from QUICKEN Java application 304 can now be seen on window 322 at workstation 320 client 326.

Referring further to Figure 4, by way of example, in order to use a Java QUICKEN financial application 304 being made available by an enterprise to anyone on the Internet on a per hour charge basis, first a telnet-based program launching tool 328 connects to the Telnet Server 318 using the sample TN5250E Telnet negotiations in connection with Figure 6. For simplicity, Telnet Server 318 uses User Exit Programs to select device name 340 QUICK0001 for this client 328. (At this point, standard negotiations are still underway.) The values of IBMRAWTADDR and IBMRAWTPORT USERVAR's are stored into LUD ASP fields 344, 348, respectively for device 308 QUICK0001 and a request issued

to initiate the Work Management process that starts the interactive client job. Work Management 338 is configured such that any devices 308 named QUICKnnnn are to be routed to special subsystem QUICKEN 314 for processing. This subsystem 314 is set up to launch the Java Virtual Machine 306 for every QUICKEN interactive job. JVM 306 connects via RAWT 312 back to the workstation RAWT port 334 using the IP address 360 and port 334 values stored in the LUD ASP fields 348, 352 for device 308 QUICK0001 and starts the QUICKEN application 304 automatically.

In accordance with alternative embodiments of the invention, many other applications in addition to QUICKEN can be offered. Further, rather than launching a single application, a menu of available Java applications can be launched. Or a menu of available languages can be launched, and so forth.

Advantages over the Prior Art

It is, therefore, an advantage of the invention that there is provided an improved system and method for enabling

graphics enabled applications to run on a text based host.

It is a further advantage of the invention that there is provided a system and method whereby an AS/400 host can enable standard Telnet Clients or Telnet-based tools to connect and be able to receive output from Java graphics applications.

It is a further advantage of the invention that there is provided a system and method for graphics enabled application platform independence for workstations by supporting a variety of clients and hardware.

It is a further advantage of the invention that there is provided an improved system and method for exploiting a Java virtual machine on a text based host system to run both text based and remote graphics applications, such as Java and X Windows applications.

It is a further advantage of the invention that there is provided a system and method for enhancing the ability of a text based host system to perform work management, including authentication, NLS, and job routing, for many clients at once.

It is a further advantage of the invention that there is provided an improved system and method for centralizing applications and support for applications.

It is a further advantage of the invention that there is provided an improved system and method for centralizing of backup/recovery processes.

It is a further advantage of the invention that there is provided an improved system and method for centralizing upgrades/fixes, such that such upgrades and fixes need be done only one time, not once for each workstation.

It is a further advantage of the invention that there is provided a system and method for comprising a single source for consulting, leasing, and marketing text based and graphical applications.

It is a further advantage of the invention that there is provided an system and method for supporting thin clients, such as network stations, by offloading CPU cycles workstations to a central mainframe.

It is a further advantage of the invention that there

is provided an improved system and method for using existing terminal emulators or Telnet-based tools, which requires no new development and exploits existing standards, including Internet RFC'S.

Alternative Embodiments

It will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without departing from the spirit and scope of the invention. In particular, it is within the scope of the invention to provide a computer program product or program element, or a program storage or memory device such as a solid or fluid transmission medium, magnetic or optical wire, tape or disc, or the like, for storing signals readable by a machine, for controlling the operation of a computer according to the method of the invention and/or to structure its components in accordance with the system of the invention.

Further, each step of the method may be executed on any

general computer, such as an IBM System 390, AS/400, PC or the like and pursuant to one or more, or a part of one or more, program elements, modules or objects generated from any programming language, such as C++, Java, Pl/1, Fortran or the like. And still further, each said step, or a file or object or the like implementing each said step, may be executed by special purpose hardware or a circuit module designed for that purpose.

Accordingly, the scope of protection of this invention is limited only by the following claims and their equivalents.